Evaluation of dependable task execution scheme for many-core systems

Tomohiro Yoneda
National Institute of Informatics

Masashi Imai
Hirosaki Univ.

Takahiro Hanyu
Tohoku Univ.

Hiroshi Saito
Univ. of Aizu

Kenji Kise
Tokyo Tech.

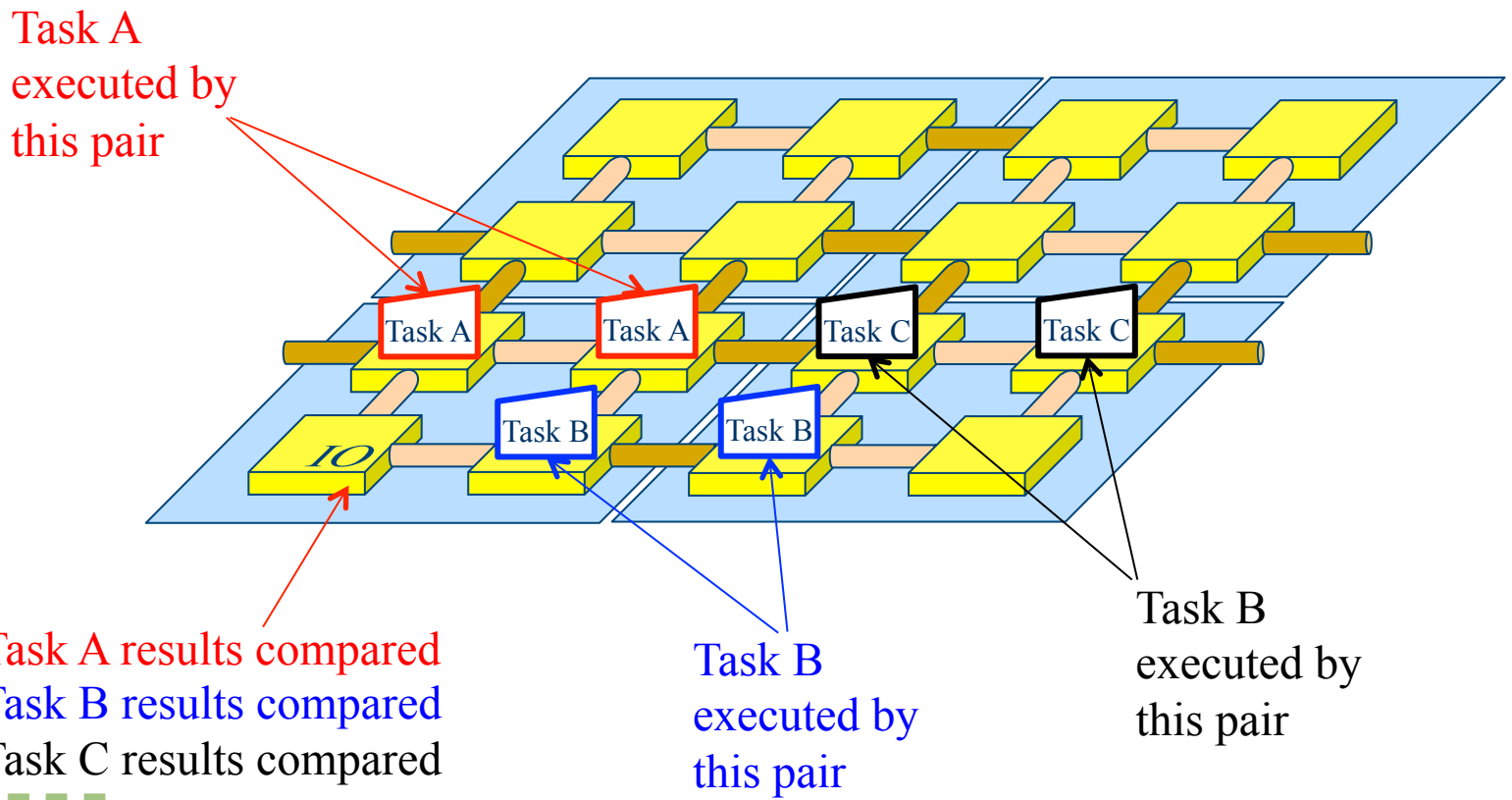
National Institute of Informatics
国立情報学研究所

Background

- Development of a many-core system to implement a centralized ECU for critical automotive applications
  - NoC based hardware
  - Dependable task execution scheme
- This progress report
  - Recent evaluation results of the dependable task execution scheme

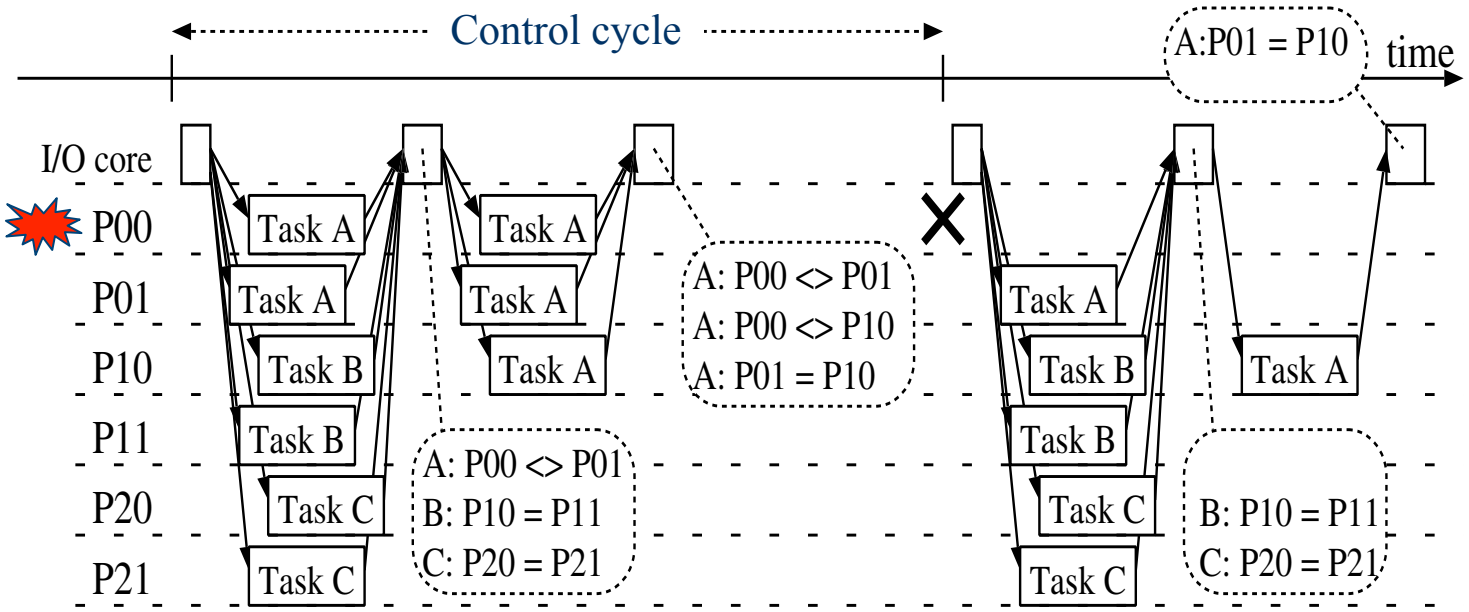
Dependable task execution scheme

- Duplicated execution, comparison, and pair-reconfiguration



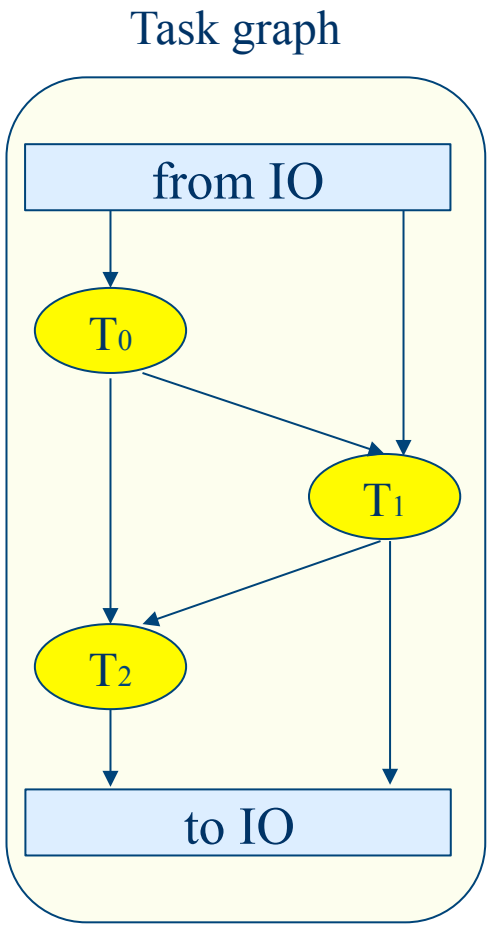
Dependable task execution scheme

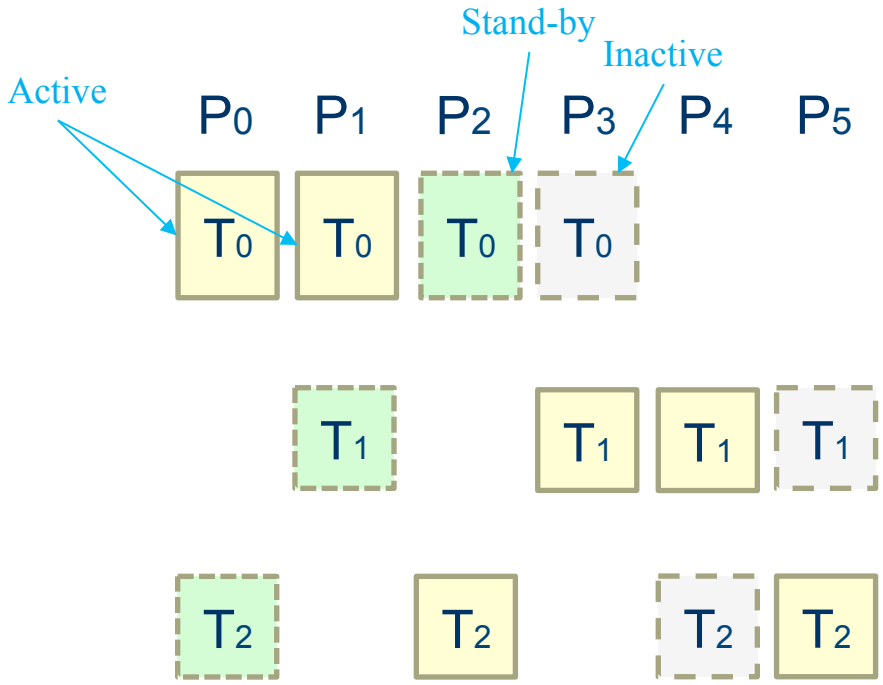
- Duplicated execution, comparison, and pair-reconfiguration



  - Active tasks are also re-executed
    - Transient errors can be masked

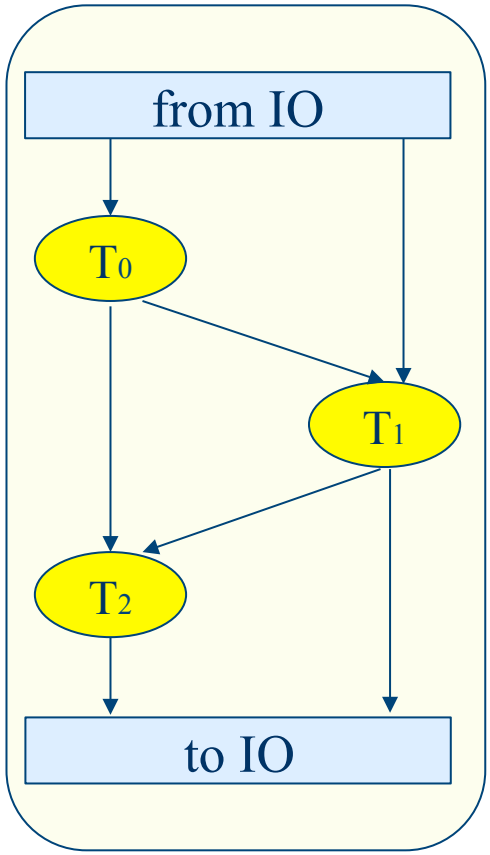
Static / Redundant Task Allocation

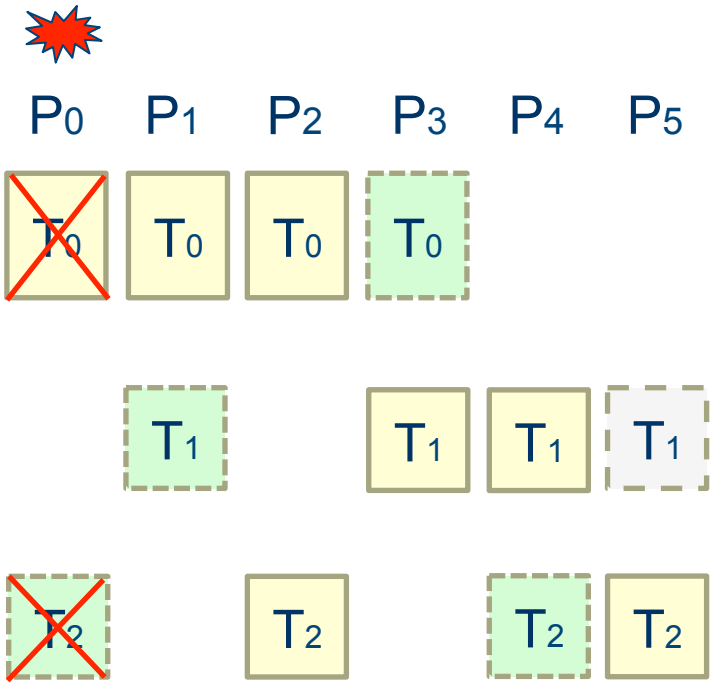




Static / Redundant Task Allocation

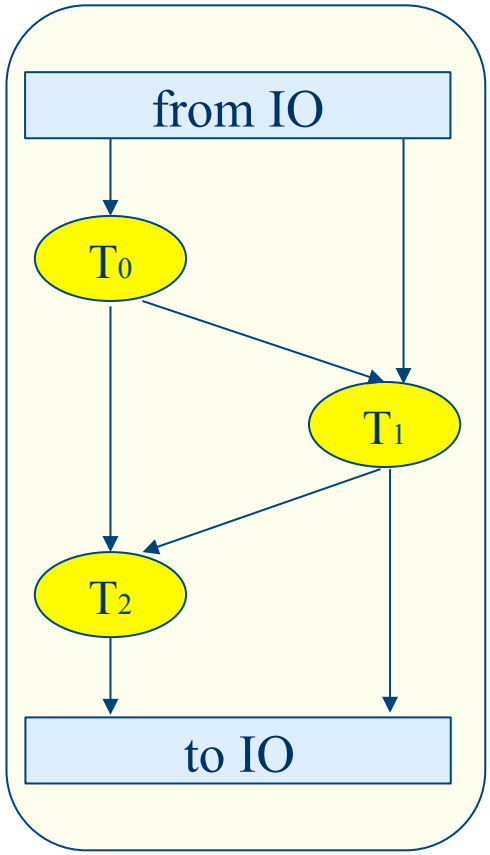
Task graph

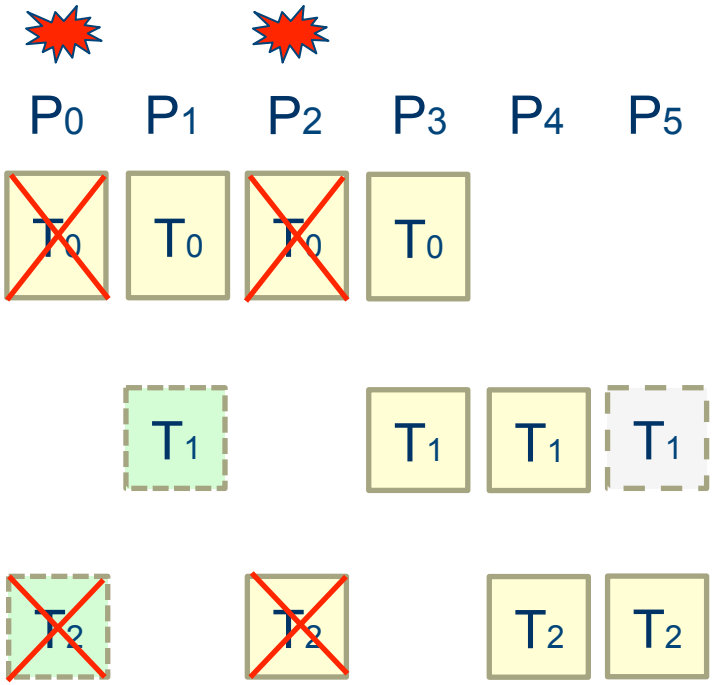




Static / Redundant Task Allocation

Task graph

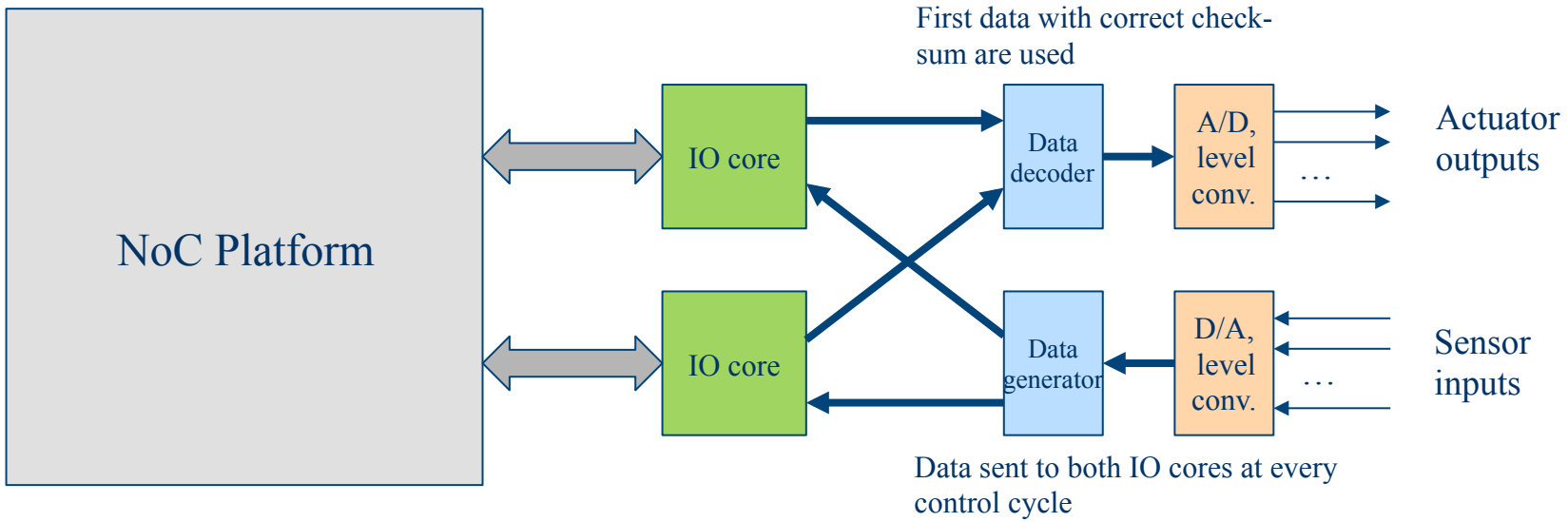




Alert should be indicated

IO core duplication

- IO core plays simple but important roles
  - Implemented by hardware or a small processor
  - Simple crash fault assumed
    - Fixed duplex configuration

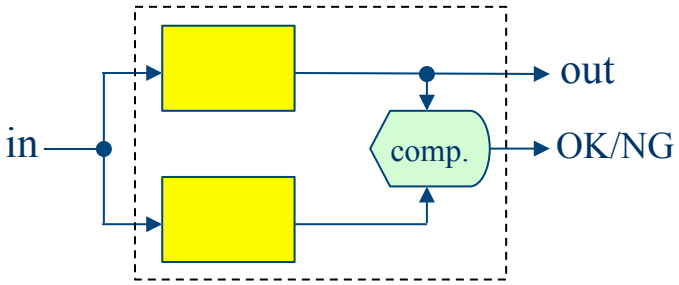


Question

- How and for what does it work better?
  - Conventional methods
    - Lock-step pair
    - TMR with a spare

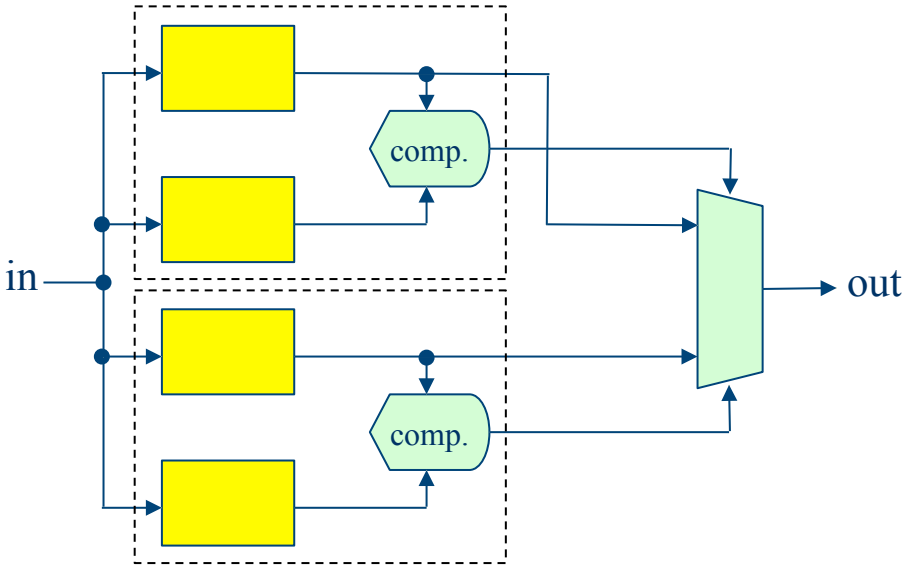
Question

- How and for what does it work better?
  - Conventional methods
    - Lock-step pair
    - TMR with a spare



Lock-step element

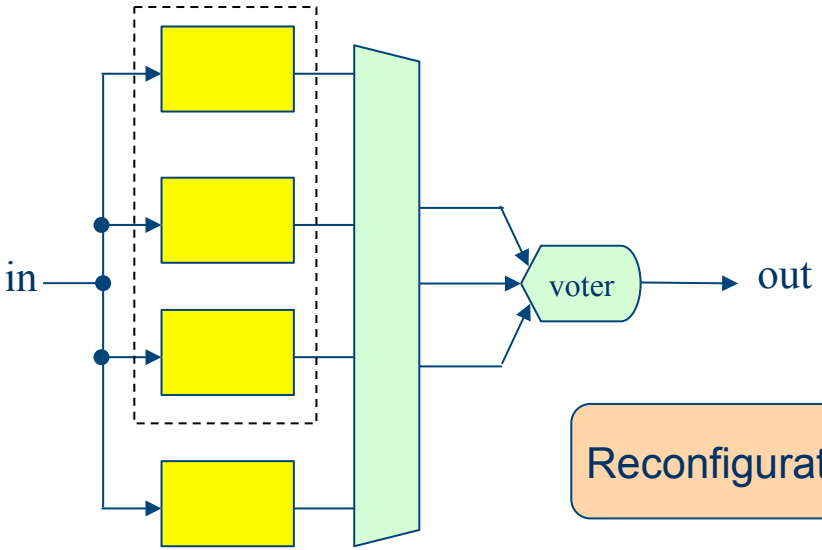
No support software needed



Lock-step pair

Question

- How and for what does it work better?
  - Conventional methods
    - Lock-step pair
    - TMR with a spare



Reconfiguration with synchronization needed

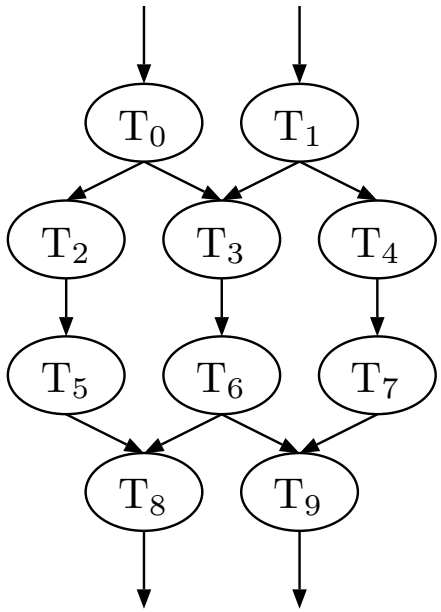
TMR with a spare

Question

- How and for what does it work better?
  - Conventional methods
    - Lock-step pair
    - TMR with a spare
- Analytical evaluation on abstracted models
  - Parameters used in this report
    - #core：8 （Failure rate λ＝10000 fit）
    - #task：10 （Execution time: T）
    - Task graph concurrency: 2-4
    - Control cycle time : CT
      (Down unless completed within CT)

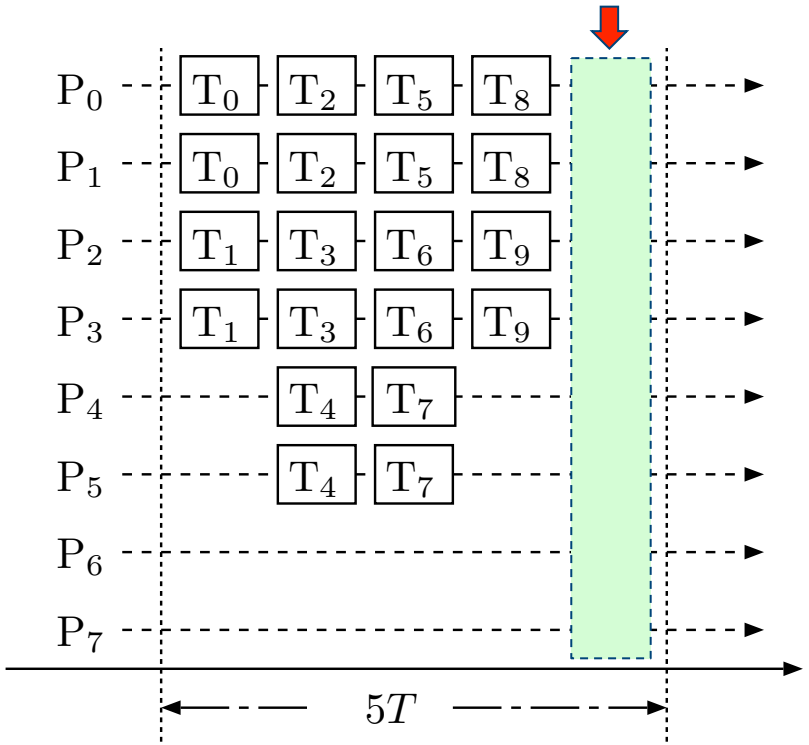
Our method

- Task graph 1
  - Concurrency: 3



(a)

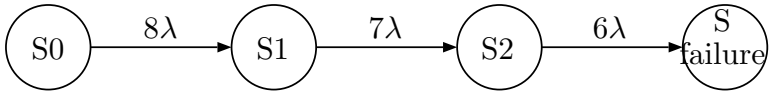
Time slot for temporary TMR

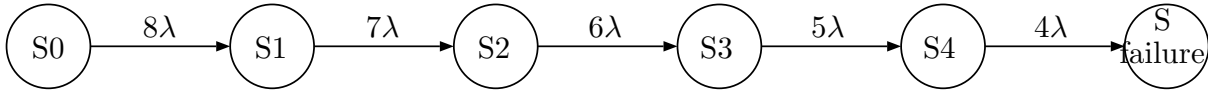


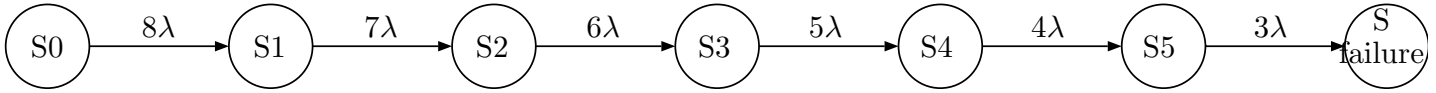
(b)

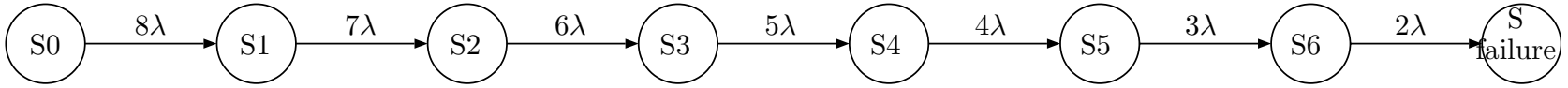
Calculation of reliability

- For given CT
  - Condition of “down” is decided
    - Eg. if CT=6T, the system goes down when 5 cores go faulty
- Markov Chains
  - 5T≤CT<6T
  - 6T≤CT<8T
  - 8T≤CT<11T
  - CT≥11T

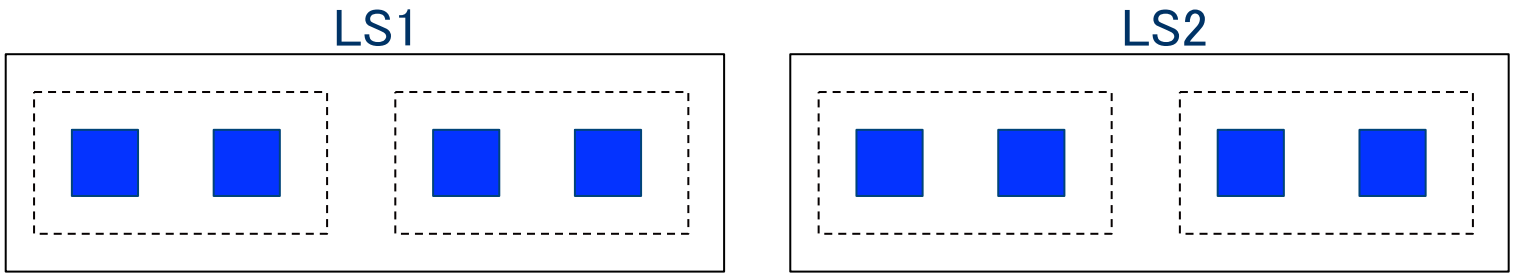








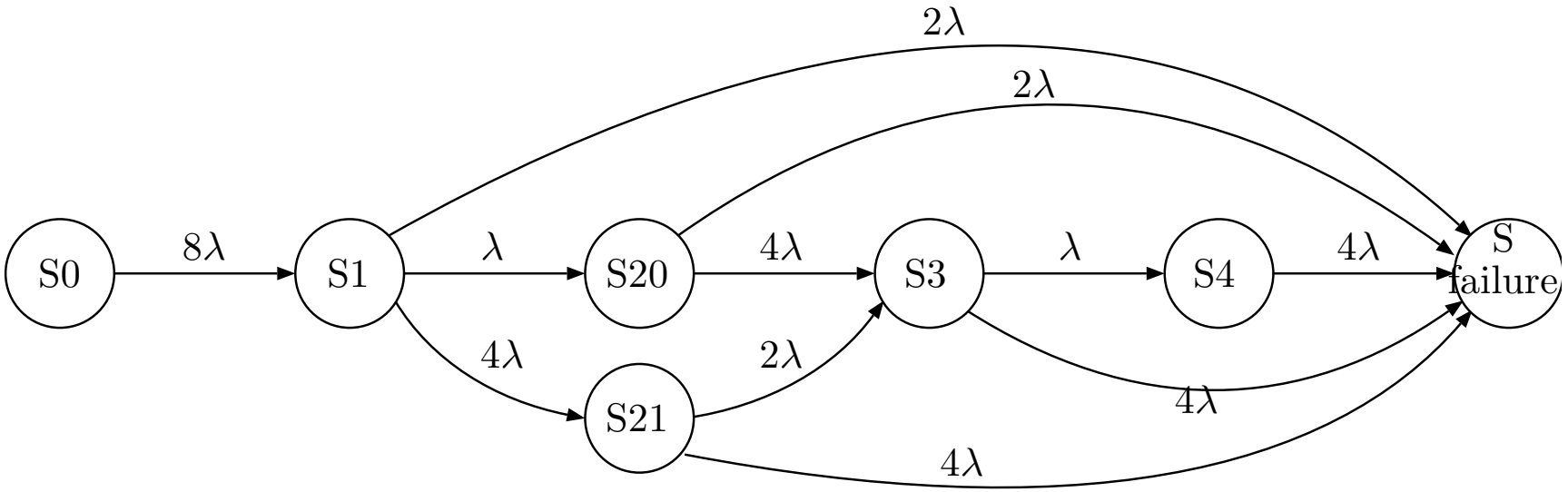
Lock-step pairs



- Type 1
  - After one fault occurs, both LS1 and LS2 are still used for the assigned tasks
- Type 2
  - After one fault occurs, only fault-free LS is used for the whole tasks

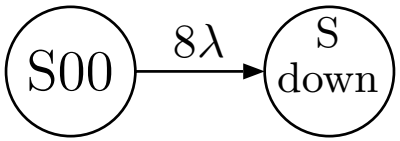
Calculation of reliability

- Markov chain for Type1
  - CT≥5T

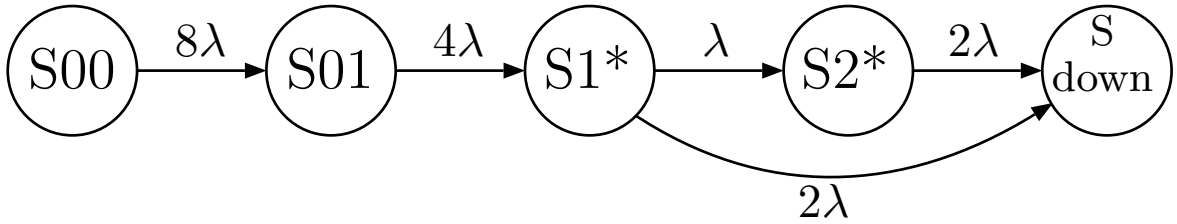


Calculation of reliability

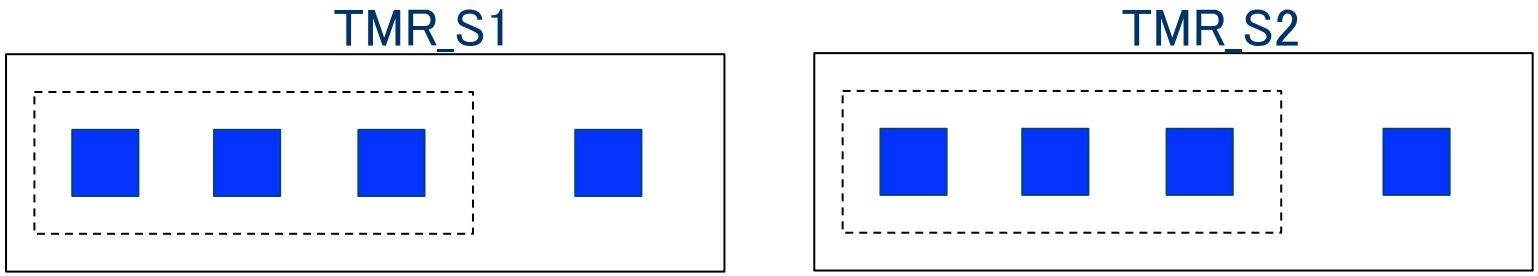
- Markov chain for Type2
  - 5T≤CT<10T



  - CT≥10T



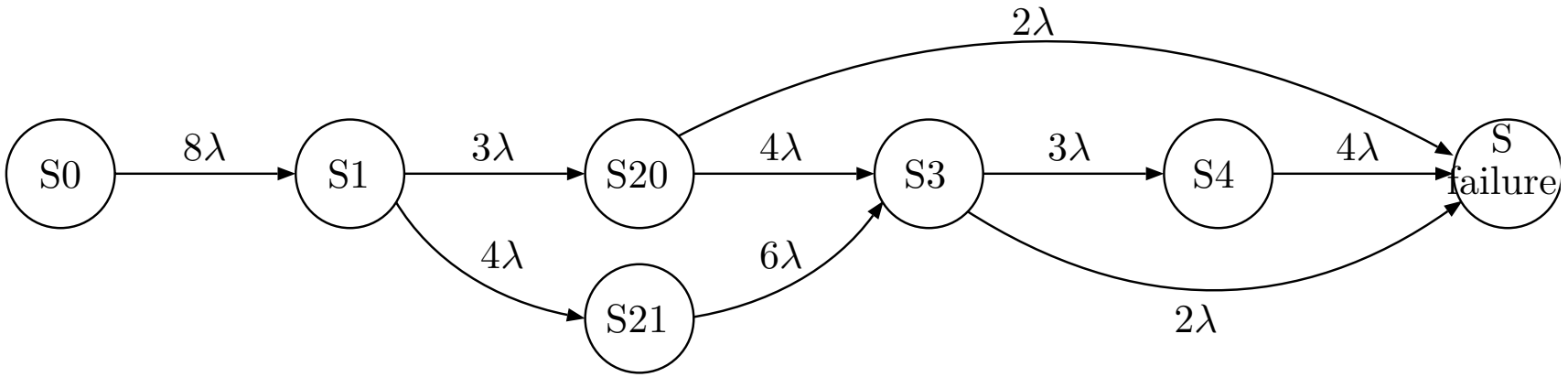
TMR with a spare



- Type 1
  - After one fault occurs, both TMR_S1 and TMR_S2 are still used for the assigned tasks
- Type 2
  - After one fault occurs, only fault-free TMR_S is used for the whole tasks

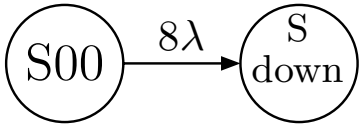
Calculation of reliability

- Markov chain for Type1
  - CT≥5T

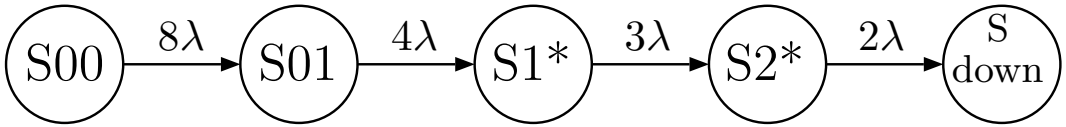


Calculation of reliability

- Markov chain for Type2
  - 5T≤CT<10T

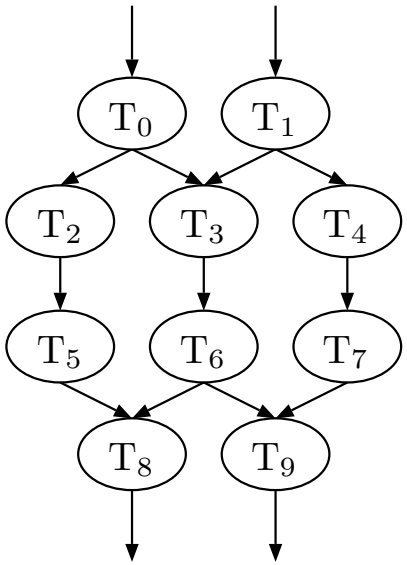


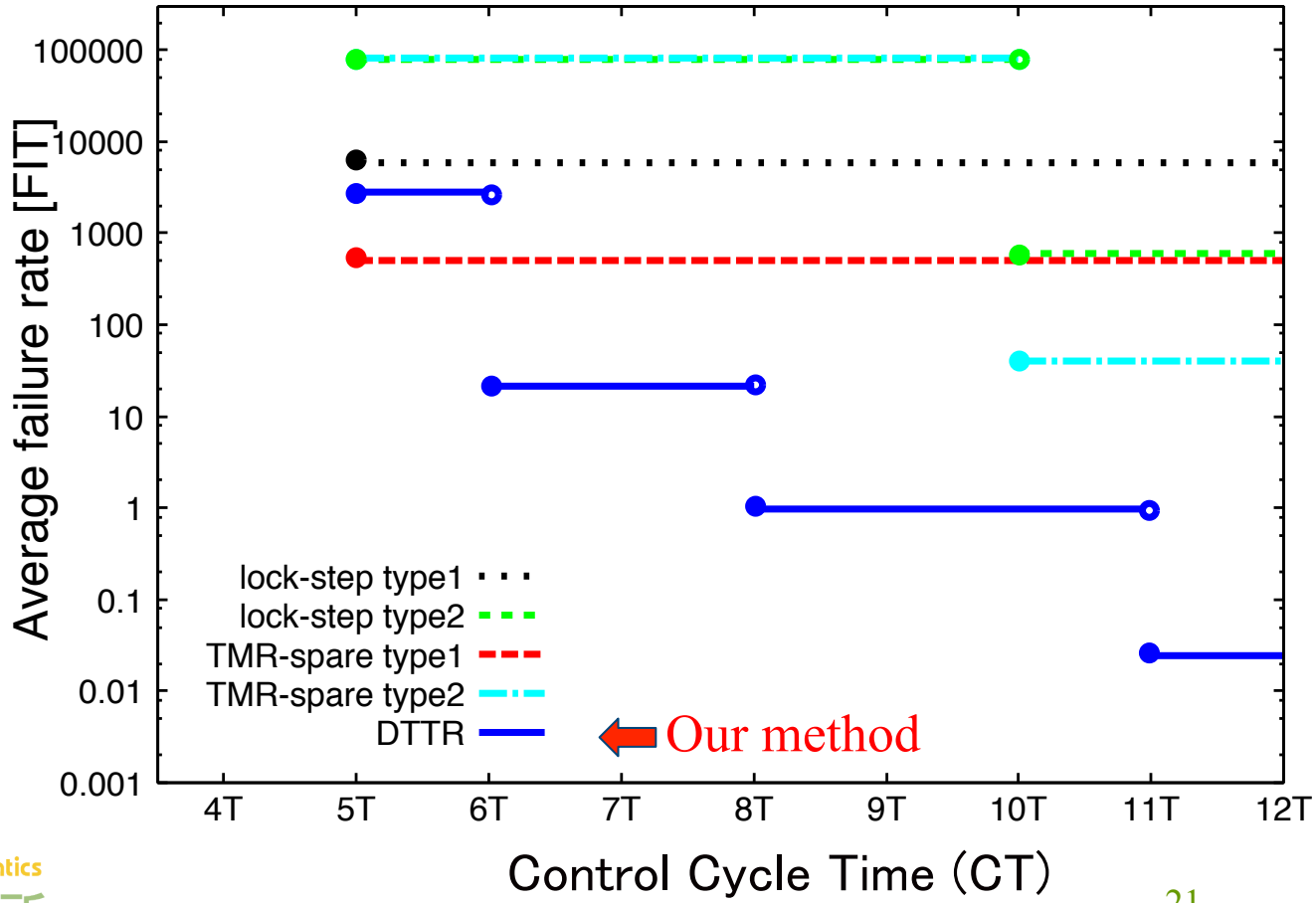
  - CT≥10T



Comparison of average failure rates

- Task graph 1



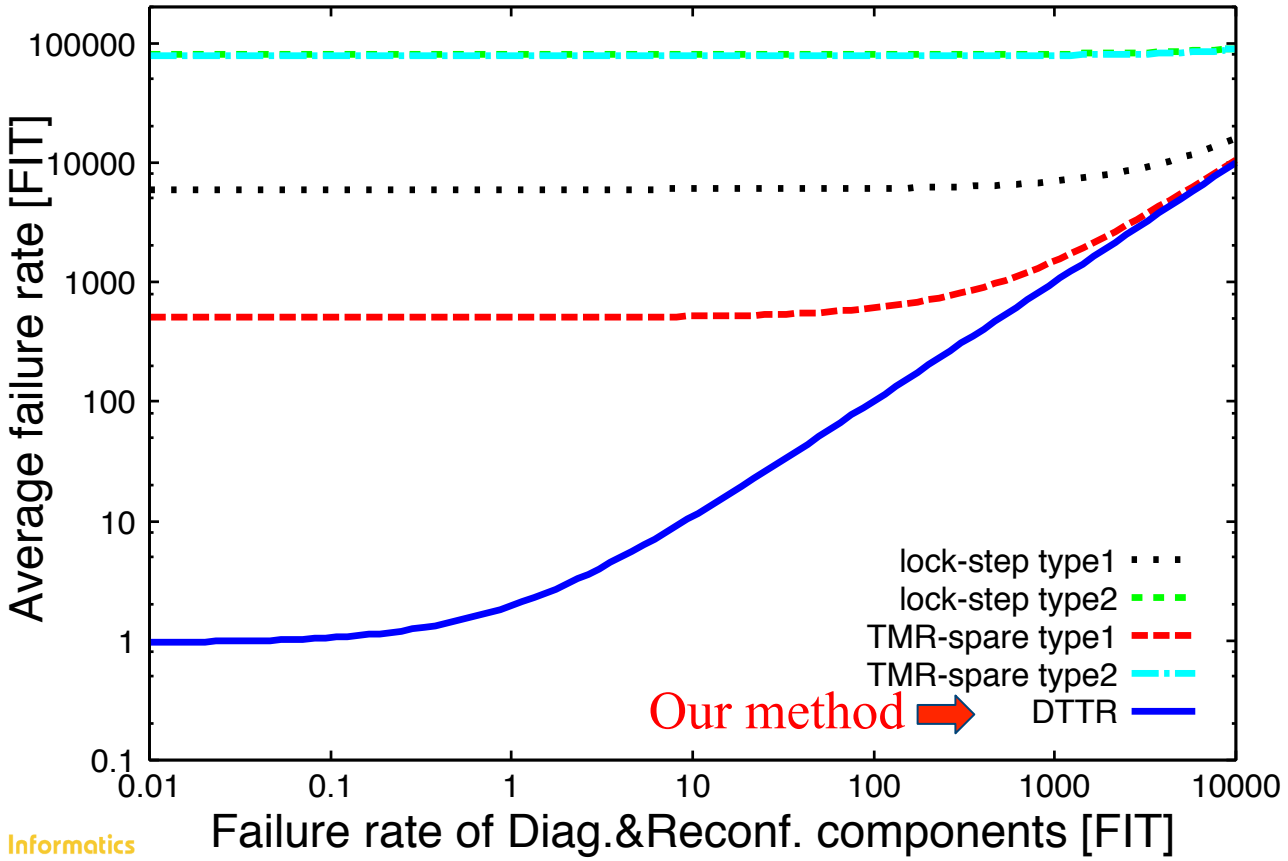


Diag. & Reconf. components

- Proposed: λ_D
  - I/O core
- LS2: λ_L
  - Comparators
  - I/O
- TMR_S: λ_T
  - Voters, Reconfiguration circuits
  - I/O
- $\lambda >> \lambda_D > \lambda_T > \lambda_L$

Diag. & Reconf. components

- For various failure rates of Diag. & Reconf. components (Task graph1, CT=8T)



Summary

- This analysis suggests
  - For large CT
    - Our method achieves highest reliability
  - With high performance cores
    - Task execution times become smaller
    - CT becomes larger relatively
- Our method has unique characteristics

Performance contributes improvement of reliability